[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/855,241	05/15/2001	Jude A. Rivers	YOR920000809US1 (14138)	8988

7590 03/19/2004

Richard L. Catania
Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, NY 11530

EXAMINER	
GERSTL, SHANE F	
ART UNIT	**PAPER NUMBER**
2183	3

DATE MAILED: 03/19/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	**Application No.** 09/855,241	**Applicant(s)** RIVERS ET AL.
	Examiner Shane F Gerstl	**Art Unit** 2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 12/18/01 and 5/15/01.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-18 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-18 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 15 May 2001 is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☒ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 2.	6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-18 have been examined.

Papers Received

2. Receipt is acknowledged of information disclosure statement paper submitted, where the paper has been placed of record in the file.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: Method and apparatus for reducing logic activity in a microprocessor using reduced bit width slices that are enabled or disabled depending on operation width.

4. Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

5. The disclosure is objected to because of the following informalities: the abstract is quite a bit larger than the 150-word maximum set forth above and lines 13 and 17 use

the terms “consisting of” and “consists of” respectively. These terms are generally accepted as claim language and should not be used in the abstract.

Appropriate correction is required.

Drawings

6. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference sign(s) not mentioned in the description: elements 201 and 202 of figure 2; elements 410 and 420 of figure 4; elements 580 and 581 of figure 5; elements 600, 691, and 692 of figure 6; and elements 130, 132, 134, 136, 138, and 139 of figure 13. A proposed drawing correction, corrected drawings, or amendment to the specification to add the reference sign(s) in the description, are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Oath/Declaration

7. The oath or declaration is defective. A new oath or declaration in compliance with 37 CFR 1.67(a) identifying this application by application number and filing date is required. See MPEP §§ 602.01 and 602.02.

The oath or declaration is defective because:
The second page of the oath and declaration is missing some characters. The first instance of the heading “Inventor’s Signature” is missing the first letter ‘I’. The last heading that should read “Post Office Address” under the entry that reads “Same as Residence” is missing.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1, 5, and 6 are rejected under 35 U.S.C. 103(a) as being unpatentable over Canal (Very Low Power Pipeline Using Significance Compression) in view of Emma (4,943,908).

10. In regard to claim 1,

a. Canal discloses a multistage microprocessor pipeline structure for executing processing instructions comprising:

i. an instruction cache (figure 1), a decoder (A RISC pipeline such as the one used here as shown on page 181, section 1 inherently has a decoder. Also, since Section 2.3 discusses the instruction formats of the instruction cache and how they are manipulated, there must be a decoder to analyze these formats from the instruction cache stage), a register file (figure 1), an arithmetic logic unit (figure 1, ALU), and a cache memory (figure 1, data cache), wherein to start execution of an instruction, the instruction is fetched from the instruction cache, and the instruction is then decoded for the operation by the decoder (which is inherent as receiving instructions from he cache), and corresponding register values for execution of the instruction are read from the register file and are input to the arithmetic logic unit which executes the instruction (all as shown in figures 1, 3, 5, 7, and 9);

ii. width determination logic receives outputs from the decoder and determines a minimum effective processing operation width for executing each processing instruction and propagates width control data along with data for execution of the instruction through the pipeline structure for executing the processing instruction; On the first page, second column, and first paragraph it is shown that a number of extension bits flow through or propagate through the pipeline (as shown in figures 1, 3, 5, 7, and 9) appended to all data to gate-off unneeded activity at each stage. Section 2.1 details that the extension bits give indication of the number of significant bytes of data that must be operated on, or a minimum effective processing operation width. There is inherently some width determination logic to determine this width data. Also, since the decoder identifies and determines how to handle the instruction formats as shown above, the data for the width determination logic would inherently be from this decoder.

iii. the microprocessor pipeline structure comprises a plurality of reduced bit width slices for execution of the instruction, wherein each slice comprises a reduced bit width portion of the register file, a reduced bit width portion of the arithmetic logic unit, and a reduced bit width portion of the cache memory, with a data carry operation proceeding from a lesser significant slice to a more significant slice, and the slices all operate in parallel when a full bit width processing operation is executed, or only a minimum required numbers of slice is enabled if the width of the processing operation is determined to be narrower than a full bit width processing operation, and different slices are enabled and process data on a cycle-by-cycle basis. Figure 9 shows a plurality of parallel reduced width bit slices for instruction execution (based on the basic pipeline of figure 1) where each slice includes reduced width portions of the register file (R0-R3), arithmetic logic unit (ALU0-ALU3), and cache memory (D0-D3). Section 2.5 shows under "Case 2" that carry-ins from preceding bytes (less significant bytes) exist and thus the data carry operation is disclosed. As shown previously, logic that is not used is turned off and thus if the width of operations are narrower than the full width available, these unused slices are turned off. This is further shown on page 189, column 2 where it says that only the functional units that operate on significant bytes are used.

b. Canal does not disclose an instruction buffer where the instructions from the instruction cache are loaded into the instruction buffer.

c. Emma has taught an instruction buffer (figure 1, element 11) that receives instructions from an instruction cache before being sent to the processor for decoding (column 2, lines 49-53).

d. Column 2, lines 54-68 show that an instruction buffer alleviates some of the bandwidth requirement on the cache and allows for instructions to be properly aligned before decoding. The abilities to alleviate some of the bandwidth requirement from the cache and to allow for proper instruction alignment would have motivated one of ordinary skill in the art to modify the design of Canal to use the instruction buffer taught by Emma.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Canal to incorporate the instruction buffer as taught by Emma so that proper instruction alignment may be realized and so that some of the bandwidth requirement on the cache may be lifted.

11. In regard to claim 5, Canal in view of Emma discloses the multistage microprocessor pipeline structure of claim 1, wherein a cache tag file stores addresses of the cache memory to write to and read from, and a width tag file stores the width of data stored in each memory address. Figures 3, 5, 7, and 9 all show that the cache memory (D) is address by a tag file. Section 2.6 shows that the extension bits are appended to each data word read and written to and from an address. Since these extension bits give the width, the storage space that holds them is the width tag file.

12. In regard to claim 6, Canal in view of Emma discloses the multistage microprocessor pipeline structure of claim 1, wherein the width determination logic outputs the width control bits to enable data flow and computation in the slices. As shown above, slices are turned on and off according to the width determination and thus the width determination logic must output control bits.

13. Claims 2-4 and 8-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Canal in view of Emma as applied to claim 1 above, and further in view of Hennessy.

14. In regard to claim 2,

a. Canal in view of Emma discloses the multistage microprocessor pipeline structure of claim 1, wherein the width determination logic uses data about the length of operands stored in a register file tags module that stores value bit information about each operand in the register file, including the width of each operand, and one or more leading bits of one or more bytes of each operand, which are examined for data overflow from a lesser significant slice to a more significant slice. Section 2.1 gives more specific description of the extension bits mentioned above that give information about the data operands. Figures 1, 3, 5, 7, and 9 all show that each stage has a module to store the extension bits or tags of the operands. The register file or operand fetch stage has this module as well and an appropriate name for it would then be a register file tags module. Page 182, column one, last paragraph shows that the extension bits (or value bit information) tell the total number of sign-extended bytes. As shown in the context of this section, this information gives the number of significant bytes in the operand, or the width.

b. Canal in view of Emma as applied to claim 1 does not disclose that the register tag module information includes the sign of each operand nor that the one or more leading bits of one or more bytes of each operand are examined for data overflow from a lesser significant slice to a greater significant slice. It is however inherent that the system identifies the sign since these sign bits are not included in the width determination as shown in section 2.1; the sign is simply not stored in the tag module.

c. Hennessy discloses in pages 221-22 that overflow occurs when the sign of the result is different than the sign of operands that both have the same sign. Since the sign is identified by the leading bits of data, examining the operands for overflow involves examining the leading bits of the operands. It would have been convenient for these signs needed for the overflow determination to be in a central location such as a bit in the tag module disclosed by Canal in view of Emma. The width determination would clearly be affected by overflow since the result is of a different width with and without overflow considerations.

d. Also, page 223 of Hennessy shows that when overflow is detected, an exception is invoked and corrective code is executed and inherently correct results are obtained. The ability to execute this correct code and get correct data using sign information would have motivated one of ordinary skill in the art to modify the design of Canal in view of Emma to include the overflow detection using the value of leading bits of operands stored as taught by Hennessy.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Canal in view of Emma to incorporate the overflow logic and methodology using the value of leading bits of operands taught by Hennessy so that correct results are obtained.

15. In regard to claim 3, Canal in view of Emma and further in view of Hennessy discloses the multistage microprocessor pipeline structure of claim 2, wherein the value bit information includes a sign of an operand in one bit, a register data width in bytes of the operand value in two bits, and one or more leading bits of one or more of the most significant bytes of the operand. As shown above, the sign is stored in a bit of the register file tag module, which stores the value bit information. Page 182, column 2, last paragraph of Canal shows that the extension bits, or data width as shown above, are two bits. The sentence directly above this paragraph shows that these extension bits are used to indicate the significance of the other three bytes. This leaves a number of leading bits in this most significant byte as value information as well since data is not specified there.

16. In regard to claim 4, Canal in view of Emma and further in view of Hennessy discloses the multistage microprocessor pipeline structure of claim 2, wherein the output of the decoder indicates two source registers and one destination register, and an instruction operation code, and the register file tags module and the instruction operation code are used to determine the number of slices required for executing the corresponding processing instruction, and those number of slices are enabled in subsequent cycles in the pipeline structure. Figure 2 of Canal gives the use of the MIPS R-format and I-format of the instructions. The R-format indicates an opcode (instruction operation code) and three registers. One of ordinary skill in the art would recognize that rs and rt are the source registers, and rd is the destination register in this MIPS format and these values are output by the decoder, which has been shown to

interpret the instructions. As shown above the data widths in the register file tag module determine the number of slices required for execution. The operation code is also used. Table 3 shows instructions such as ADDU (add unsigned) and SLL (shift left logical), which at times generate a carry to another slice (which would need to be enabled) because of their function. Instructions such as XOR do not generate a carry and thus there is no need to enable another slice. Since these instructions are specified by the instruction operation code one recognizes that this operation code has an impact on the number of slices required for execution.

17. In regard to claim 8, Canal in view of Emma and further in view of Hennessy discloses the multistage microprocessor pipeline structure of claim 2, wherein the width determination logic determines the likelihood of a data overflow being generated from a narrow slice operation by examining one or more leading bits of the operands which are stored in the register file tags module and generates one of three determinations: no data overflow is guaranteed, and the effective operation width is determined by the width of the narrow operands; data overflow is guaranteed, and the effective operation width must be one byte larger than the width of the narrow operands; data overflow is possible but not certain, wherein a carry into the bits examined is propagated as a carry out. As shown above, overflow affects width and thus is used for width determination. Also, the sign bits, which are the values of the leading bits, are stored in the register file tags module. The overflow determination decides if overflow occurs or not, both possibilities being one of the three options claimed in the alternative. The width determination given above enables needed and disables unneeded slices. If there is

overflow in a slice, inherently another slice is enabled to receive the carry of the shifted sign bit, which is also supported above. If there is no overflow, then no extra slices need to be enabled.

18. In regard to claim 9,

a. Canal in view of Emma discloses the multistage microprocessor pipeline structure of claim 1, wherein following execution and completion of a processing operation by the arithmetic logic unit, the width of the value of the processing operation result is determined. The top of section 2.5 shows that the ALU produces significant result bytes as well as the extension bits (value width) that go with them. Also, figure 1 shows a unit "G" after the ALU. Column 2 of page 181 shows that these G points are where the extension or width bits are determined. Canal in view of Emma also show in figure 1 that results are written back to a destination register in the register file.

b. Canal in view of Emma does not disclose that, after which the width determination logic determines value bit information for the processing operation result by combining its sign bit, value width and one or more leading bits for its one or more leading bytes. It is however inherent that the system identifies the sign since these sign bits are not included in the width determination as shown in section 2.1; the sign is simply not stored together with the value width and leading bits.

c. Hennessy discloses in pages 221-22 that overflow occurs when the sign of the result is different than the sign of operands that both have the same sign.

Since the sign is identified by the leading bits of data, examining the operands for overflow involves examining the leading bits of the operands. It would have been convenient for these signs (which are the leading bits) needed for the overflow determination to be in a central location with other pertinent data disclosed by Canal in view of Emma. The width determination would clearly be affected by overflow since the result is of a different width with and without overflow considerations.

d. Also, page 223 of Hennessy shows that when overflow is detected, an exception is invoked and corrective code is executed and inherently correct results are obtained. The ability to execute this correct code and get correct data using sign information would have motivated one of ordinary skill in the art to modify the design of Canal in view of Emma to include the overflow detection using the value of leading bits of operands stored as taught by Hennessy.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Canal in view of Emma to incorporate the overflow logic and methodology using the value of leading bits of operands taught by Hennessy so that correct results are obtained.

19. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Canal in view of Emma as applied to claim 1 above, and further in view of Brooks (Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance).

20. In regard to claim 7,

a. Canal in view of Emma discloses the multistage microprocessor pipeline structure of claim 1,

b. Canal in view of Emma does not disclose wherein enabling and disabling of each slice is accomplished by clock gating, where during enabling, clock signals allow data to proceed into and through a slice, and during disabling, clock signals block the flow of data into and through a slice.

c. Brooks has disclosed on page 1, column 2 the use of clock gating to disable functional units that will not be used for an operation. This technique would be advantageous for disabling the slices of Canal in view of Emma that will not be used.

d. Brooks has taught in this same section that clock gating significantly reduces power consumption. This reduction in power consumption would have motivated one of ordinary skill in the art to change the disclosed invention of Canal in view of Emma to use clock gating to disable unused slices as taught by Brooks.

It would have been obvious to modify the design of Canal in view of Emma to use the clock gating technique taught by Brooks for disabling unused slices so that power consumption us significantly reduced.

21. Claims 10-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Canal in view of Moline (4,941,119).

22. In regard to claim 10,

a. Canal discloses a method for reducing logic activity in the execution of an operation in a processor comprising the steps of:

i. selecting at least one operand associated with said operation, looking up a width and a value of selected bits of said at least one operand, determining an effective width of said operation based upon the width of said at least one operand, a function specified by said operation enabling the width of the resources in said processor corresponding to said effective width of said operation for executing said operation. Section 2.1 shows that the operands have extension bits that indicate the significant bits and thus the width of significant data or the effective width is determined using these bits so that unneeded activity is gated off as shown on page 181, column 2. Section 2.1 shows that this effective width is found by looking up the data and width of the operand and encoding the number of significant bits. The enabling is further shown on page 189, column 2 where it says that only the functional units that operate on significant bytes are used.

ii. executing said operation (figure 1, via the ALU)

iii. determining the width of the result of said operation based upon the step of executing. The beginning of section 25 shows that the ALU produces significant result bytes as well as the extension bits (width) that go with them. Also, figure 1 shows a point "G" after the ALU. Column 1 of page 181 shows this to be a point where the widths or extension bits are determined.

b. Canal does not disclose determining a prediction of arithmetic overflow, based upon the width and the value of said selected bits of said at least one operand, where said prediction of arithmetic overflow is basis for the effective width.

c. Moline has taught a method for prediction of integer multiply (arithmetic) overflow based on the value and width of the operand. The summary shows that the prediction takes place by finding the most significant bit of the operands and summing them and comparing that to the width of the result register. Column 3, lines 43-57 shows an example and that finding the most significant bit of the operand is finding it's width since if the operand MSB (which is based on the operand value) is in the 21st place, the operand is effectively 21 bits long just as in Canal. Therefore, the prediction is based on the width and value of the operands. The width determination would clearly be affected by overflow since the result is of a different width with and without overflow considerations.

d. Column 1, lines 22-32 show that Moline's disclosure is a substantially faster implementation that overflow detection. The ability to predict overflow in parallel with execution and not waste time with overflow detection would have motivated one of ordinary skill in the art to modify the design of Canal to include the design taught by Moline.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Canal to incorporate the overflow prediction used by Moline so that overflow is dealt with in a quick and efficient manner.

23. In regard to claim 11, Canal in view of Hennessy discloses the method of claim 10, including saving the width of the result of said operation, and saving said result of said operation. Figure 1 of Canal shows that the result of the ALU is saved in the data cache and the register file destination. The figure also shows that the extension bits (width) of the result are sent back to the register file as well.

24. In regard to claim 12, Canal discloses the method of claim 10, wherein said step of looking up a width and a value of selected bits includes dedicated hardware for holding and retrieving said width and said value of selected bits. Figure 1 of Canal shows that there are dedicated pipeline registers for holding the data operands between stages. And dedicated "exten" registers for holding the width or extension bits.

25. In regard to claim 13, Canal in view of Hennessy discloses the method of claim 10, wherein the processor includes a register file (figure 1), an arithmetic unit (figure 1, ALU), a memory path (figure 1, cache fill from main memory and paths to and from instruction and data caches), and a cache memory (figure 1, data cache), and the register file, the arithmetic unit, and the cache memory are divided into a plurality of slices, each of which is of a reduced bit granularity, and the bits in all of the slice form a full width word in the processor. Figure 9 shows a plurality of parallel reduced width bit slices for instruction execution (based on the basic pipeline of figure 1) where each slice includes reduced width portions of the register file (R0-R3), arithmetic logic unit (ALU0-

ALU3), and cache memory (D0-D3). The figure shows four slices and the beginning of section six shows this full parallelism to be 4 byte (8-bit) parallelism. Therefore, four slices of four bytes yields a full word.

26. In regard to claim 14, Canal in view of Hennessy discloses the method of claim 13, wherein at least one slice is of 8 bit granularity. As shown above, the slices are 8-bits wide each.

27. In regard to claim 15, Canal in view of Hennessy discloses the method of claim 13, wherein at least one slice is of 16 bit granularity. Paragraph 2 of section 2.1 of Canal shows that in some embodiments slices are 16-bits (halfword granularities).

28. In regard to claim 16, Canal in view of Hennessy discloses the method of claim 13, wherein said step of enabling includes logic to enable a required number of slices to execute the operation. As shown in section 1, portions of the pipeline (slices) that are not in use are disabled and thus the number required for use are enabled.

29. In regard to claim 17,

a. Canal discloses a processor comprising:

i. a plurality of slices, each of which is a portion of a full width word of the processor, wherein each slice comprises a portion of a register file, a portion of functional units, a portion of a memory path, a portion of a cache memory, and a portion of other resources required to perform operations in the processor. Figure 9 shows a plurality of parallel reduced width bit slices for instruction execution (based on the basic pipeline of figure 1) where each slice includes reduced width portions of the register file (R0-

R3), functional units (ALU0-ALU3), cache memory (D0-D3), and memory path (cache fill is from main memory and the paths to and from the caches). The figure shows four slices and the beginning of section six shows this full parallelism to be 4 byte (8-bit) parallelism. Therefore, four slices of four bytes yields a full word.

ii. logic to save and retrieve a width and selected bits of operands used to perform an operation in the processor; Figure 9 shows "exten" registers for holding and retrieving from other stages extension bits. The extension bits are shown in section 2.1 to give the amount of significant data and thus the width of data for operation.

iii. logic to determine a number of slices required to perform the operation based upon the width of one or more operands, the functionality of the operation, and the prediction of arithmetic overflow, Section 1, column 2 shows that unneeded portions (or slices) of the pipeline are gated off or disabled based on the extension bits or the width needed as the required number slices is determined.

iv. logic to activate the slices required to perform the operation. As shown above the number of slices is determined and unused slices are turned off, thus, the used slices are inherently enabled. This is further shown on page 189, column 2 where it says that only the functional units that operate on significant bytes are used.

b. Canal does not disclose logic to determine a prediction of arithmetic overflow when performing the operation, based upon the width and the selected bits of the operands used to perform the operation,

c. Moline has taught a method for prediction of integer multiply (arithmetic) overflow based on the value and width of the operand. The summary shows that the prediction takes place by finding the most significant bit of the operands and summing them and comparing that to the width of the result register. Column 3, lines 43-57 shows an example and that finding the most significant bit of the operand is finding it's width since if the operand MSB (which is based on the operand value) is in the 21st place, the operand is effectively 21 bits long just as in Canal. Therefore, the prediction is based on the width and value of the operands. The width determination would clearly be affected by overflow since the result is of a different width with and without overflow considerations.

d. Column 1, lines 22-32 show that Moline's disclosure is a substantially faster implementation that overflow detection. The ability to predict overflow in parallel with execution and not waste time with overflow detection would have motivated one of ordinary skill in the art to modify the design of Canal to include the design taught by Moline.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Canal to incorporate the overflow prediction used by Moline so that overflow is dealt with in a quick and efficient manner.

30. In regard to claim 18, Canal discloses the processor of claim 17, including logic to determine the width of the result of the operation (Page 181, column 2, paragraph 1 shows that the extension bits and thus the width is determined at points "G," where one of these places is shown in figure 1 for the result of the ALU), circuitry to store said width and selected bits of said result (figure 1, pipeline register from this G point and "exten" register), and circuitry to store said result (pipeline registers, data cache, register file all store the result).

Conclusion

31. The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

32. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. The following patents and publication have been cited to further show the art with respect to division of the pipeline into slices in general.

US Pat No 6,192,384 to Dally shows a system with slices comprising ALU clusters and local registers.

US Pat No 5,010,511 to Hartley teaches bit slices comprising full adders that send a carry to each other slice from a least significant to most.

"Efficiency of microSIMD architectures and index-mapped data fro media processors" to Ruby B. Lee discusses slices that include 16-bit functional units and portions of a register file.

"Multimedia Extensions for General-Purpose Processors" to Ruby B. Lee discloses using subword parallelism for multimedia applications and how ALU's are partitioned and read from certain parts of a register file.

"Multimedia Processor-Based Implementation of an Error-Diffusion Halftoning Algorithm Exploiting Subword Parallelism" to Jae-Woo Ahn teaches partitioning of the ALU.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Shane F Gerstl whose telephone number is (703)305-7305. The examiner can normally be reached on M-F 6:45-4:15 (First Friday Off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703)305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Shane F Gerstl
Examiner
Art Unit 2183

SFG
March 17, 2004

[illegible]
EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100